Convoy Movement Problem: A Civilian Perspective

Azar Sadeghnejad-Barkousaraie, Rajan Batta, Moises Sudit

Department of Industrial and Systems Engineering, University at Buffalo (State University of New York), Buffalo, NY 14260, USA

Revised March, 2016
Originally Submitted March, 2015

Abstract

We study the convoy movement problem in peacetime from a civilian perspective by seeking to minimize civilian traffic disruptions. We develop an exact hybrid algorithm that combines the k-shortest path algorithm along with finding a minimum weighted k-clique in a k-partite graph. Through this coupling scheme, we are able to exactly solve large instances of the convoy movement problem without relaxing many of its complicating constraints. An experimental study is performed based on pseudo transportation networks to illustrate the computational viability of the method as well as policy implications.

Keywords: Convoy Movement Problem, Disjoint path, k-shortest path, k-clique

1 Introduction and Motivation

The Convoy Movement Problem (CMP) is a special case of the vehicle routing and scheduling problem in which vehicle length cannot be neglected and in which there are special rules regarding convoy encounters on the network. To better understand this we present some details about convoys and their study from the operations research perspective. A convoy is a chain of (mostly large) vehicles which need escort via security units. Each convoy needs to be dispatched from a source node to a specific destination node while meeting due date constraints. Convoys may have different speeds on each road type of the network. Convoys are commonly used by the military to transfer ammunition, food, medicines, clothing and military or marine forces to other bases or sea ports.

Paths followed by different convoys do not have to be necessarily disjoint but two convoys may not occupy the same part of the network at the same time (Lee et al. 1996), called a no crossing or blocking constraint. In addition, convoys cannot overtake each other on the network. A minimum headway should be maintained between two convoys passing through the same road in the same direction, to avoid accidents. And once a convoy starts its movement, it must continue moving until it reaches its destination (Chardaire et al. 2005), called a no halting constraint. Some researchers address the Peacetime Convoy Movement Problem (PCMP) by relaxing the no halting constraint (Kumar and Narendran 2010). Our consideration of the peacetime problem is not because of these halting periods (which can be handled by adding a loop to any specific resting point to the network) but by consideration of disruption to civilian traffic life caused by convoy movement.

What distinguishes convoy routing from regular vehicle routing problems, is that convoys are large compared to the length of each road segment on the network, sometimes taking one hour to pass through an intersection (Marine Corps 2005). As a result, civilian traffic may be disrupted dramatically by convoy movement. Also, convoys need additional escort cars that ensure that traffic is clear for movement on streets or highways (Marine Corps 2005). We study the CMP from the civilian point of view. Since there is no pressing security issue, we don't need to send a convoy through the shortest possible path, as long as it meets the arrival deadline at its destination. We aim instead to minimize the traffic disruption impact to civilians. We assume that the no halting constraint is still active, to preserve a minimum level of security for each convoy. Our research offers a hybrid exact method to solve this problem, while preserving the problem's complicated features. We assume a discrete departure time, so a convoy cannot depart its origin at any time after the earliest ready time, but has to wait a specified time interval to commence its movement.

The rest of the paper is organized as follows: Section 2 contains a review of the CMP literature. Section 3 presents a mathematical formulation of the problem and its variations. Section 4 details our proposed solution approach. Section 5 contains results from our computational experiments. Finally, section 6 provides a summary, conclusion and future research opportunities.

2 Literature Review

Chardaire et al. (2005) were the first to prove that the CMP is NP-complete, by showing that the disjoint connecting path problem can be polynomially reduced to the decision version of CMP. Gopalan and Narayanaswamy (2009) studied a special case of CMP when convoys have zero length, and established that it is NP-complete by polynomially reducing the 3-satisfiability problem to this restricted version of CMP.

CMP has been widely studied in the literature while considering time related objective functions. There is still no efficient exact algorithm for solving the general case of the CMP. One of the first formal studies of CMP in recent years began through the paper by Bovet et al. (1991). They studied a convoy scheduling problem in which a set of different types of convoys, with specific time windows, share the same road as part of their paths to their destinations. The problem they tackled determines how and when a convoy will join and leave this road while all CMP constraints are satisfied. They proposed a mixed integer mathematical model, proved that this specific traffic problem is NP-complete and represented a Tabu search procedure to solve this problem. Lee et al. (1996) solved a general version of CMP by using three different methods, Pure Branch and Bound Algorithm(B&B), hybrid Genetic Algorithm (GA) with B&B, and pure GA. However in their proposed B&B approach there is no guarantee that paths generated for each convoy are simple.

Tuson and Harrison (2005) used a random search algorithm, based on a modified Dijkstra's algorithm, to solve real cases of CMP. Robinson and Leiss (2006) used agent based simulation to construct the hierarchy of a convoy formation's vehicle units, columns, serial units and march units, and then used genetic algorithm to solve the convoy scheduling problem to find a conflict free solution. Kumar and Narendran (2011) proposed a procedure for finding lower bounds for CMP using Lagrangean relaxation by relaxing overtaking and blocking constraints, under the circumstance where there is no time limit enforced on convoys. Gopalan (2015) studied the computational complexity of CMP under different restricted constraints and presented polynomial-time algorithm for two-particle convoys movement problem, where convoy length and overtaking constraints were relaxed.

Table 1: Summary Table of CMP Literature Review.

Article	**Blocking**	**Overtaking**	**Time Constraint**	**Heterogeneous Convoys**	**Exact**	**Special Constraint**	**Objective**
Our Work	Yes	Yes	Yes	Yes	Yes	Tabu list for some parts of the network	**Total Traffic Disruption**
Bovet et al. (1991)	Yes	Yes	Yes	No	Yes (MILP)	One road	Total Completion Time
Lee et al. (1996)	Yes	Yes	Yes	Yes	Yes	Not simple Path	Total Completion Time Max Completion Time
Montana et al. (1999)	No	No	Yes	No	No	Single O-D Pair	Joint Vehicle Capacity Balance Route Capacity Total Travel Time Number of Switches of Roads
Harrison (2001)	Yes	Yes	Yes	No	No	No	Total Completion Time
Tuson and Harrison (2005)	Yes	Yes	Yes	No	No	No	Total Completion Time Max Completion Time
Chardaire et al. (2005)	Yes	No	Yes	Yes	No	No	Total Completion Time
Robinson and Leiss (2006)	Yes	Yes		No	No	Not optimization	NA
Gopalan and Narayanaswamy (2009)	Special	No	No	Yes	No	zero length convoys capacity on nodes	Total Completion Time
Thangarajoo and Lau (2010)	Yes	No	Yes	No	No	No	Total Completion Time
Goldstein et al. (2010)	Special	Yes	No	Yes	No	Road disaster and limitation	General Cost Quantity
Kumar and Narendran (2010)	Yes	Yes	Yes	No	Yes (MILP)	Zero length convoy	Total Completion Time
Kumar and Narendran (2011)	Yes	Yes	No	Yes	No	No	Total completion time
Gopalan (2015)	Yes	Yes	Yes	Yes	No	Yes	Total completion time Makespan Total traveled distance

Because of difficulty of solving the general case of the CMP, researchers often present the general case of the CMP but focus only on solving special cases. We now briefly describe these special cases along with a synopsis of the methodology used.

Montana et al. (1999) used genetic algorithms for convoy formation and routing problem with a single O-D pair problem. Harrison (2001) used lagrangian relaxation on the conflict prevention constraints and repeatedly used Dijkstra's algorithm to solve the problem in a time-space network, to minimize total traveling time and maximum completion time. Chardaire et al. (2005) used lagrangian relaxation in a time-space network to solve a special case of CMP when there are no overtaking constraints by assuming that the speed of all convoys are the same. Gopalan and Narayanaswamy (2009) studied the on-line CMP with zero length (particle convoys) and extended convoys, where the demand for each convoy increases dynamically in time. Thangarajoo and Lau (2010) used a hybrid heuristic method based on information sharing to solve the CMP by considering each convoy as an agent with a set of jobs. Goldstein et al. (2010) used a genetic algorithm to solve the CMP with a universal speed limit on the graph and a special encountering constraint of convoys at intersections. Kumar and Narendran (2010) provide a classification scheme for CMP and proposed a robust integer programming model for the zero length CMP. They also presented two heuristic based approaches for a simplified version of CMP with no headway and blocking constraints.

What distinguishes our work is that we simultaneously enforce several realistic constraints, and through a solution methodology that utilizes concepts from the k-shortest path and the k-clique problem are able to preserve problem solvability. Table 1 summarizes the CMP literature and places our work in the context of this literature.

3 Mathematical Formulation

In this section mathematical formulations of the general problem and its variations are provided.

3.1 Assumptions

. The transportation network is represented by a directed graph $G = (N, A)$, where N is a set of nodes or intersections and A is a set of arcs or roads.

. Each convoy has a specified physical length, which consists of the length of each vehicle and the safety gap between each vehicle inside the convoy. The safety gap for each vehicle is three times the length of the vehicle immediately in front of it.

. Each convoy needs to be sent from an origin to a specific destination.

. Each convoy has an earliest departure time from its origin, and a latest arrival time to its destination or due date.

. Convoys should not overtake each other on the same road.

. Two convoys cannot face each other when traversing in opposite direction of the same road.

. Convoys can only wait at their origin for discretely defined time intervals. Waiting is not permitted elsewhere.

Table 2: Indices, Parameters and Decision Variables.

		Explanation
Indices	i, j, i', j'	Graph nodes $\in N = \{1, 2, ..., M\}$
	k, k'	Convoys $\in \{1, 2, ..., K\}$
	v	Convoys vehicles $\in \{1, 2, ..., v_k\}$
Parameters	K	Number of convoys or O-D pairs
	M	Number of nodes on the network (intersections)
	$l_{i,j}$	Length of the arc $(i, j) \in A$
	l_k^v	Length of the vehicle v in convoy k
	v_k	Number of vehicles in convoy k
	o_k, d_k	Origin and Destination points of convoy k
	H	Headway between two convoys passing the same node
	r_k	Earliest ready time (departure time) from its origin for convoy k
	f_k	Latest arrival time to its destination for convoy k
	$S_{i,j}^k$	Speed of convoy k on arc $(i, j) \in A$
	$F_{i,j}$	Traffic volume on arc $(i, j) \in A$
	$A_{A\times A}$	Tabu List matrix; corresponding arcs should not be simultaneously occupied at the same time by more than one convoy
	l_k	Total length of convoy k
	λ_k	Waiting time interval at the origin for convoy k
Decision Variables	$x_{i,j}^k$	$\begin{cases} 1, & \text{If convoy } k \text{ will pass through arc } (i, j) \in A \\ 0, & \text{Otherwise} \end{cases}$
	$y_{k,k'}^i$	$\begin{cases} 1, & \text{If convoy } k \text{ will pass node } i \text{ before convoy } k' \\ 0, & \text{Otherwise} \end{cases}$
	$z_{i,j}^k$	The time interval which convoy k is on the arc $(i, j) \in A$
	$t_i^{k,v}$	The time when v^{th} vehicle of convoy k arrives at node i
	$q_{i,j,i',j'}^{k,k'}$	$\begin{cases} 1, & \text{if Convoy } k \text{ will pass the arc } (i, j) \in A \text{ before convoy } k' \text{ passes arc } (i', j') \in A \\ 0, & \text{Otherwise.} \end{cases}$
	c_i^k	The time when convoy k will arrive to node i
	b_k	The integer waiting component of convoy k

. The speed of each convoy on each road of the transportation network is constant and may vary from convoy to convoy, and road to road. This speed is the average speed of the slowest vehicle in each convoy, usually the first vehicle of each convoy (Marine Corps 2005).

. There is a specified set of roads that we do not want simultaneously occupied by convoys. In order to impose this assumption, we define a tabu list with these specified roads.

3.2 General Mathematical Model

A list of indices, parameters and decision variables are presented in Table 2. We now present the problem formulation without the tabu list constraints. These are added later in section 3.3.

$$(\mathcal{P}) \quad \min \sum_k \sum_{(i,j)\in A} F_{i,j} z_{i,j}^k \tag{1}$$

$$\sum_{i=1}^{m} x_{i,j}^k - \sum_{j'=1}^{m} x_{j,j'}^k = \begin{cases} 1 & j = d_k \\ -1 & j = o_k \\ 0 & otherwise \end{cases} \qquad \forall j, \forall k \tag{2}$$

$$\sum_{i=1}^{m} x_{i,j}^{k} \leq 1 \qquad \forall k, \forall j \tag{3}$$

$$\sum_{j=1}^{m} x_{i,j}^{k} \leq 1 \qquad \forall k, \forall i \tag{4}$$

$$t_i^{k,1} + l_{i,j}/S_{i,j}^{k} + M(1 - x_{i,j}^{k}) \geq t_j^{k,1} \qquad \forall k, \forall (i,j) \in A \tag{5}$$

$$t_i^{k,1} + l_{i,j}/S_{i,j}^{k} - M(1 - x_{i,j}^{k}) \leq t_j^{k,1} \qquad \forall k, \forall (i,j) \in A \tag{6}$$

$$t_{o_k}^{k,1} \geq r_k \qquad \forall k \tag{7}$$

$$t_{d_k}^{k,v_k} + l_{v_k}^{k}/S_{i,d_k}^{k} \leq f_k + M(1 - x_{i,d_k}^{k}) \qquad \forall k, \forall v, \forall i \tag{8}$$

$$t_i^{k,v} + 3 \times l_k^{v}/S_{i,j}^{k} + M(1 - x_{i,j}^{k}) \geq t_i^{k,v+1} \qquad \forall k, \forall v, \forall (i,j) \in A, i \neq d_k \tag{9}$$

$$t_i^{k,v} + 3 \times l_k^{v}/S_{i,j}^{k} - M(1 - x_{i,j}^{k}) \leq t_i^{k,v+1} \qquad \forall k, \forall v, \forall (i,j) \in A, i \neq d_k \tag{10}$$

$$t_{d_k}^{k,v} + 3 \times l_k^{v}/S_{j,d_k}^{k} - M(1 - x_{j,d_k}^{k}) \leq t_{d_k}^{k,v+1} \qquad \forall j \neq d_k, \forall k, \forall v \tag{11}$$

$$t_i^{k,v_k} + H - M(1 - y_{k,k'}^{i}) \leq t_i^{k',1} \qquad \forall k, k', \forall i \tag{12}$$

$$\sum_{i=1}^{m} x_{i,j}^{k} + \sum_{i'=1}^{m} x_{i',j}^{k'} \geq 2 \times (y_{k,k'}^{j} + y_{k',k}^{j}) \qquad \forall j \neq o_k, o_{k'}, \forall k, k', k \neq k' \tag{13}$$

$$1 + \sum_{i=1}^{m} x_{i,j}^{k} + \sum_{h=1}^{m} x_{h,j}^{k'} \geq 2 \times (y_{k,k'}^{j} + y_{k',k}^{j}) \qquad j = o_k \text{ or } j = o_{k'} \forall k, k, k \neq k' \tag{14}$$

$$(y_{k,k'}^{j} + y_{k',k}^{j}) \leq 1 \qquad j = o_k \text{ and } j = o_{k'}, \forall k, k', k \neq k' \tag{15}$$

$$\sum_{i=1}^{m} x_{i,j}^{k} + \sum_{h=1}^{m} x_{h,j}^{k'} \leq y_{k,k'}^{j} + y_{k',k}^{j} + 1 \qquad \forall j \neq o_k, o_{k'} \forall k, k', k \neq k'; \tag{16}$$

$$M(2 - x_{i,j}^{k} - x_{j,i}^{k'}) \geq y_{k,k'}^{j} + y_{k',k}^{i} - 1 \qquad \forall (i,j), (j,i) \in A, \forall k, k', k \neq k' \tag{17}$$

$$1 + \sum_{i=1}^{m} x_{i,j}^{k} + \sum_{h=1}^{m} x_{h,j}^{k'} \leq y_{k,k'}^{j} + y_{k',k}^{j} \qquad j = o_k \text{ or } j = o_{k'} \forall k, k', k \neq k' \tag{18}$$

$$y_{k,k'}^{j} + y_{k',k}^{j} \geq 1 \qquad j = o_k \text{ and } j = o_{k'}, \forall k, k' (k \neq k') \tag{19}$$

$$t_j^{k,v_k} + l_{v_k}^{k}/S_{i,j}^{k} - t_i^{k,1} \leq z_{i,j}^{k} + M(1 - x_{i,j}^{k}) \qquad \forall k \forall (i,j) \in A \tag{20}$$

$$x_{i,j}^{k}, y_{k,k'}^{i} \in \{0,1\} \qquad \forall i, j \forall k, k'$$

$$t_i^{k} \geq 0 \qquad \forall i, k$$

The objective function (1) is to minimize the total traffic disruption, calculated by the total time each convoy occupies an arc, weighted by traffic volume. (2) is a flow conservation constraint to assure continuity in the sequence of arcs a convoy traverses. Constraints (3) and (4) enforce that each convoy can leave and enter a node at most once. Constraints (5) and (6) compute the arrival time of the head of a convoy based on the selected arc. They ensure that the arrival time of the head of the convoy to a node j from arc (i, j) is the arrival time to node i, plus the time needed for the convoy to pass arc (i, j). Constraints (7) and (8) ensure that convoy k starts its trip after its earliest ready time and completes its arrival to its destination before its due date. Constraints (9),(10) and (11) calculate the arrival time of each vehicle of a convoy to each node. Constraint (12) assures that headway is maintained between two convoys which pass through the same node, such that two convoys do not occupy the same node at the same time. The set of constraints (13)

to (19) impose the blocking and overtaking restrictions. Where constraints (14) and (15) replace constraint (13), and constraints (18) and (19) substitute constraint (16) for origin node of convoy k or k'. Constraint (20) measures the amount of time that an arc is occupied by a convoy.

We now discuss variations of the general mathematical model $(\mathcal{P})$.

3.3 Incorporating an arc Tabu List

The formulation $(\mathcal{P})$ while ensuring that convoys do not "collide" does not specify any spatial separation. Thus there could be cases where convoys come close to one another and thereby increase civilian traffic disruption of an area dramatically. To keep convoys spatially separated it is useful to define an arc tabu list. The idea of the tabu list is to prevent any two arcs on the list to be occupied at the same time by different convoys. To accomplish this we define a binary matrix $A_{E\times E}$. If $A((i,j)(i',j')) = 1$, then arcs (i,j) and (i',j') cannot be occupied by two separate convoys at the same time. We need two new constraints to enforce this case:

$$t_i^{k,1} \geq t_{j'}^{k',v_k} + M(x_{i,j}^k + x_{i',j'}^{k'} - 2) \qquad \forall (i,j),(i',j') \in A, \forall k,k' \tag{21}$$

$$t_j^{k,v_k} \leq t_{i'}^{k',0} - M(x_{i,j}^k + x_{i',j'}^{k'} - 2) \qquad \forall (i,j),(i',j') \in A, \forall k,k' \tag{22}$$

Constraint (21) will be active if convoy k' passes arc $(i'j')$ before convoy k passes arc (i,j). Constraint (22) is for the reverse situation. However, both cases cannot simultaneously occure. To ensure that only one of these constraints is active we define a control variable $q_{i,j,i',j'}^{k,k'}$ for each pair of (21) and (22) constraints. The required constraints are therefore:

$$t_i^{k,1} \geq t_{j'}^{k',v_k} + M(x_{i,j}^k + x_{i',j'}^{k'} - 2) - Mq_{i,j,i',j'}^{k,k'} \tag{23}$$

$$t_j^{k,v_k} \leq t_{i'}^{k',1} - M(x_{i,j}^k + x_{i',j'}^{k'} - 2) + M(1 - q_{i,j,i',j'}^{k,k'}) \tag{24}$$

3.4 Constant Length for Each Convoy

In the formulation $(\mathcal{P})$ the length of a convoy is based on the number of vehicles assigned to it and keeps track of each vehicle's passage time through a node. However, enforcement of no "collision" constraints can also be achieved by viewing a convoy as a single entity. This simplifies the formulation without sacrificing its generality. To accommodate this change, we use c_i^k instead of $t_i^{k,v}$. Here c_i^k is the time that the head of convoy k enters node i. The length of convoy k is represented by l_k. We replace $t_i^{k,1}$ by c_i^k in constraints (5), (6) and (7). Constraints (9) and (10) can now be eliminated and constraints (8), (12) are modified respectively as follows:

$$c_{d_k}^k + l_k/S_{i,d_k}^k \leq f_k + M(1 - x_{i,d_k}^k) \qquad \forall k$$

$$c_j^k + l_k/S_{i,j}^k + H - M(2 - y_{k,k'}^j - x_{i,j}^k) \leq c_j^{k'} \qquad \forall k,k',\forall (i,j) \in A$$

In addition, $z_{i,j}^k$ can be replaced by $\frac{l_{i,j}+l_k}{S_{i,j}^k} x_{i,j}^k$, because by constraint (20), if $x_{i,j}^k = 1$, $z_{i,j}^k = c_j^k - c_i^k + l_k/S_{i,j}^k$, which by constraint (5) $c_j^k - c_i^k = \frac{l_{i,j}}{S_{i,j}^k}$. Constraint (20) can be eliminated and objective function is updated as follows:

$$\min \sum_k \sum_{(i,j)\in A} \frac{l_{i,j}+l_k}{S_{i,j}^k} F_{i,j} x_{i,j}^k \tag{25}$$

3.5 Imposing Waiting Interval at the Origin

The formulation ($\mathcal{P}$) allows complete flexibility in departure time for a convoy. For analytic tractability (i.e. the exact solution method presented in section 4 we restrict departure time to be at discrete intervals. Specially, we assume that the departure time of convoy k is $r_k + b_k\lambda_k$ where b_k is the number of discrete-length intervals of length λ_k which convoy k waits at its origin o_k. To accommodate this we therefore change constraint (7) to:

$$t_{o_k}^{k,1} = r_k + b_k\lambda_k \qquad \forall k$$

4 Exact Solution Methodology (Best Choice Algorithm)

We propose an exact approach for solving CMP, by decomposing it into smaller sub-problems (finding k-shortest paths with minimum average traffic volume for each convoy) and then combining solutions of those sub-problems into one minimum weighted k-clique in k-partite graph problem to find a promising feasible solution. Decomposition approaches helps to solve problems with large size or high complexity by solving large number of smaller sub-problems (Vanderbeck and Savelsbergh 2006). In order to decomposing our problem to smaller sub-problems, we need to first define the concept of compatible paths.

Definition 1. Compatible Paths. Two paths are compatible if their combination do not violate blocking, overtaking and arc-tabu list constraints. Paths for the same O-D pair are considered incompatible, regardless of their departure times. We show the compatibility of path p_1 and path p_2 by $p_1 \equiv p_2$ representation.

If the size of a path i is shown by $|P_i|$ the complexity of checking the compatibility of a pair of paths (i, j) is $O(min(|P_i|, |P_j|)) < O(V)$, because we need to check each edges (and its endpoint) of one the paths once, and check if the other path contains the same arc, its tabu-arcs or its endpoint, by choosing an efficient data structure, this comparison for each vertex will take $O(1)$ time. In the worst case scenario we need to check all arcs in one path, the size of each simple path in a graph with V number of nodes is at most $(|V - 1|)$, so at most it will take $O(V)$ time.

Our solution methodology melds two algorithms to address this decomposition and find the optimal solution. At first, for each convoy k-shortest paths (minimum average traffic volume) are generated and only feasible paths (due date constraint) are kept, as sub-problems. Each sub-problem separately, generates best possible paths for each convoy, by considering release time, due date and no halting constraint (constraints from (2) to (11) in section 3.2). Which is the same as finding the shortest path on a graph with arc weights(i, j) set to $\frac{l_{i,j}+l_k}{S_{i,j}^k} F_{i,j}$.

The second algorithm is a Branch and Bound (B&B) approach to find a minimum weighted k-Clique in a k-partite graph, to combine solutions from sub-problem. These two algorithms are implemented sequentially and iteratively, and at the end of each iteration the optimality condition is checked. Figure 1 shows the structure of the proposed method, which we call the *Best Choice* algorithm. The master problem combines all linking constraints of different paths into the compatibility definition (constraints from (12) to (19) and constraints (23) and (24)).

Since the k-Clique problem is NP-hard, we try to keep the number of vertices in the graph as small as possible. Doing so allows the proposed B&B algorithm to find the optimal solution faster than the original Branch and Bound model to solve CMP.

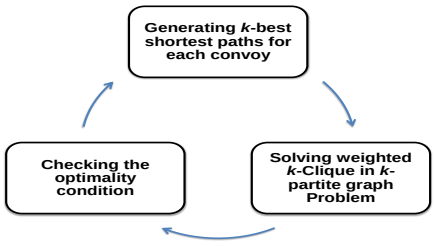

Figure 1: Proposed Approach (Best Choice algorithm).

4.1 Generating k-Best Shortest Paths for Each Convoy

The first phase of the proposed method used the k-shortest path algorithm. The aim of the k loopless shortest paths algorithm is to find k best paths for a specific origin-destination pair, sorted in a list. An excellent review of k-shortest path algorithms is available in a paper by Mohanta and Poddar (2012). We therefore briefly highlight the major contributions in this area.

Pollack (1961) presented a method for finding the k^{th} shortest path, which relies on calculating the $(k-1)$ shortest paths, setting the cost of each arc in each previously found shortest path to infinity, and then finding the shortest path from the relabeled network. The runtime of this algorithm increase exponentially with the value of k. Dreyfus (1969) and Martins and Dos Santos (1996) used label correcting algorithms (Bellman 1958, Ford Jr 1956, Moore 1959) for finding the k-shortest path. The efficiency of their algorithm is better than those of brute force algorithms.

The Optimality principle asserts that "there is a shortest path formed by shortest sub-paths", and can be further extended for k-shortest path problem. Yen's algorithm (Yen 1971) is designed with direct implementation of this assertion. The algorithm starts with the best shortest path as the first one. Then to generate the second shortest path, the first shortest path will be used to find a node, as "deviation node", and the second shortest path will find another path from this node to the destination, by not using nodes in the previous path. The algorithm continues the process of utilizing previous paths to compute the new path.

Further improvements in computational complexity are observed with the Martins and Dos Santos (1996) algorithm and the new implementation of Yen's algorithm proposed by Eppstein (1998).

$O(kn+m\log m)$ is the reported improved computational complexity of the newer implementation of Yen's algorithm (Martins and Pascoal 2003). With slight modification, we used this improved version of the Yen's algorithm to find k-shortest paths in this paper. For each convoy h, a modified graph is defined. In this graph, the length of each road is updated by $\frac{l_{i,j}+l_h}{S^h_{i,j}}F_{i,j}$, where $l_{i,j}$ is the length of the road (mile), l_h is the length of the convoy h (mile), $S^h_{i,j}$ is the speed of convoy h on road (i,j) (mile per hour) and finally $F_{i,j}$ is the traffic volume of road (i,j) (number of cars per hour). The k-shortest path algorithm (Martins and Pascoal 2003) is implemented on this updated graph. We refer to the set of generated paths as $BestPath_h$.

Paths which are not feasible based on the due date are eliminated from the set $BestPath_h$. By assigning different departure time, more paths are generated from each single feasible path and added to $BestPath_h$. If no feasible path is found by running k-shortest path algorithms, we rerun the algorithm and take the paths number $k+1$ to $2k$ to generate feasible paths. The pseudo-code for generating $BestPath_h$ for each convoy is represented in Algorithm 1.

Algorithm 1 Generating k-best shortest paths for h

Input: Convoy h, o_h, d_h, number k, f_h(due date), λ_h .
Output: $BestPath_h$ (A set of sorted shortest paths from o_h to d_h)

```
ShortestPath ← k best shortest path in modified graph, with departure time 0;
for all Path i in ShortestPath do
    time_i ← the time convoy h takes to reach its destination (d_h);
    if time_i < f_h then
        j = 0;
        while j < f_h − time_i do
            Add Path p_l with departure time j to BestPath_h;
            j += λ_h;
        end while
    end if
end for
```

At the first iteration, for all convoys Algorithm 1 should be implemented, but in further iterations, this procedure is applied only for the convoy which plays the most significant role on the value of the lower-bound (LB).

4.2 Solving Weighted k-Clique in k-Partite Graph Problem

A feasible solution to the proposed CMP problem is a set of paths, one for each convoy, which are pairwise compatible, based on Definition 1. In the second phase of our method, we seek a feasible solution by finding a set of compatible paths among those generated in the first phase as explained in section 4.1. It is well known that the compatibility of objects considered by pairwise comparison can be modeled using the graph-theoretic notion of Cliques (Grünert et al. 2002). This problem can therefore be defined as follows:

$$(\mathcal{Q}) \quad \min \sum_i \sum_j x_i^j D_i^j \tag{26}$$

$$\sum_j x_i^j = 1 \qquad \forall i \tag{27}$$

$$x_i^j + x_{i'}^{j'} \leq 1 \qquad \forall i, i', j, j', \text{where } i \neq i', P_i^j \not\equiv P_{i'}^{j'} \tag{28}$$

Here P_i^j refers to path j of convoy i and x_i^j is its associated decision variable, which is 1 when path j of convoy i is selected and zero otherwise and D_i^j is the cost of this path. The objective function in (26) is to find a collection of paths with minimum cost. Constraint set (27) ensures that one path is chosen for each convoy, and constraint set (28) ensures that no two incompatible paths are chosen at the same time.

Problem ($\mathcal{Q}$) is equivalent to finding the minimum weighted k-clique in a k-partite graph $G = (V, E)$, where V is a set of vertices and E set of edges. Each vertex represents a path, and the collection of feasible paths for each convoy constitutes one partite of a graph. At the end we have a k-partite (k is the number of convoys) graph. There is a link between each pair of vertices, from different partiets, if they are compatible. Even though this problem is NP-complete there are several algorithms which can solve this problem in reasonable time (Mirghorbani and Krokhmal 2013, Vassilevska 2009). We modify the branch and bound algorithm offered by Mirghorbani and Krokhmal (2013), because it is easy to implement and fast enough for our problem. The detailed explanation of the Branch and Bound algorithm is as follows:

Since the k-Clique problem is NP-hard, we try to keep the number of vertices in the graph as

small as possible. Doing so allows the proposed B&B algorithm to find the optimal solution faster than the original Branch and Bound model to solve CMP.

4.2.1 Branch and Bound (B&B) algorithm.

The input of the B&B is a list of all feasible generated paths (vertices) for each convoy which will be updated on each branch, we call it *AllCompatibleOptions*, a lower and upper bound. The traversing strategy of the B&B algorithm is depth first.

Definition 2. AllCompatibleOptions. The list *AllCompatibleOptions* keeps an array of sorted paths for each convoy, which are compatible with currently selected paths.

Branching Among all convoys which have at least one path in *AllCompatibleOptions* list, we branch on a convoy that has the minimum number of paths. After the first iteration, the first branch is for a convoy for which new paths have been generated. While there is a path associated with this convoy on the list, we assign it to the chosen convoy in order. After a path is assigned to a convoy, *AllCompatibleOptions* are updated.

Updating *AllCompatibleOptions* When a path is selected for convoy k, all other paths of convoy k are deleted. Also deleted are paths of convoy $l(l \neq k)$, which are not compatible with the selected path for convoy k. Any update of *AllCompatibleOptions* in a lower level of the tree does not affect the corresponding list at an upper level.

Bounding The lower-bound of each branch is updated as follows. The first component of the lower-bound is the summation of objective values for all convoys that have paths assigned to them. The second component is the summation of the best possible path for each unassigned convoy. To find the best possible path for an unassigned convoy, we identify the best compatible path. We note that the lower-bound cannot decrease at any stage, since selection of a path for a specific convoy cannot decrease the objective value of the best compatible path for unassigned convoy.
The upper-bound can only be updated when a better feasible solution is found.

Fathoming A branch is fathomed under the following conditions:

1. *Due to upper-bound:* This happens when the value of the upper-bound is better (less) or equal to the value of the lower bound.
2. *Due to feasible solution:* At the very bottom level (level K) of the tree, when the objective value is less than the current upper-bound, all corresponding branches at the $K-1$ level are fathomed.
3. *Due to Infeasible solution:* When the *AllCompatibleOptions* is updated and the number of convoys which have at least one path inside the list is less than the current level of the tree. It means there is no feasible path for at least one convoy, based on the current selection of paths, so this branch is not feasible.

The pseudo-code of the proposed B&B algorithm is shown in Algorithm 2.

4.3 Checking the Optimality Condition

Definition 3. Optimality Condition. The Optimality Condition is defined as a situation in

Algorithm 2 B&B algorithm for k clique in k-partite graph

Input: All feasible paths of each convoy, generated by k-best shortest path algorithm, upper-bound, lower-bound.
Output: Updated lower-bound, upper-bound, possible updated feasible solution.

```
lower-bound ← all shortest path vallues;
upper-bound ← Values of the worst scenario;
AllCompatibleOptions ← all generated feasible paths ;
CurrentSelectedPath ← {};
Level ← 1;
Branch(AllCompatibleOptions, lower-bound, upper-bound, Level)
    if Level == N then
        CurrentSelectedPath add First path in AllCompatibleOptions for N;
        Update upper-bound ;
        Fathom by 2;
    else
        h ← Convoy with the fewest number of paths in AllCompatibleOptions;
        for all Path p_h in AllCompatibleOptions of h do
            CurrentSelectedPath add p_h;
            Update AllCompatibleOptions list;
            if AllCompatibleOptions is empty for non-assigned convoy then
                Fathom by 3;
            end if
            Update lower-bound;
            if lower-bound≥upper-bound then
                Fathom by 1;
            end if
            Level ← Level + 1;
            Branch(AllCompatibleOptions, lower-bound, upper-bound, Level);
        end for
    end if
```

which a feasible solution is found and adding more paths will not result in a better solution.

In order to perform the optimality check, an overall lower-bound at the end of each B&B algorithm needs to be calculated. Suppose that there are N convoys in the system, the algorithm is in its k^{th} iteration and the best feasible solution (incumbent) found so far is S; S consists of an assignment of paths to each convoy. Consider the situation where m_i feasible paths are generated for convoy i. The first one (referred as V_i) has the best value (least total traffic disruption) and the last path (referred as W_i) has the worst value. Under such a circumstance, checking the optimality condition is as follows:

Definition 4. Best Case Scenario for a Convoy. Best case scenario or B_i is the minimum(best) objective function value, after generating more paths for convoy i. B_i is calculated as follows:

> If an additional path is created for convoy i, its objective value is at best equal to W_i. In the best situation, this path would be compatible with best paths of other convoys, V_j, where $j \neq i$, so $B_i = \sum_{j(j\neq i)}\{V_j + W_i\}$. If the value of the upper-bound (Objective function value of incumbent solution) is less than B_i, no new paths needs to be generated for convoy i, because it will not improve the current best solution. A schematic representation of this concept is shown in Figure 2.

The overall lower-bound is the minimum value of all of these best case scenarios($LB = Min_i\{B_i\}$). And for the next iteration of the algorithm, just for convoy j where $j = \arg\min_i\{B_i\}$, extra k-shortest paths are generated.

If the value of the current upper-bound, is less than the general lower-bound value, it means that there is no better feasible solution for the problem and the optimal solution has been found.

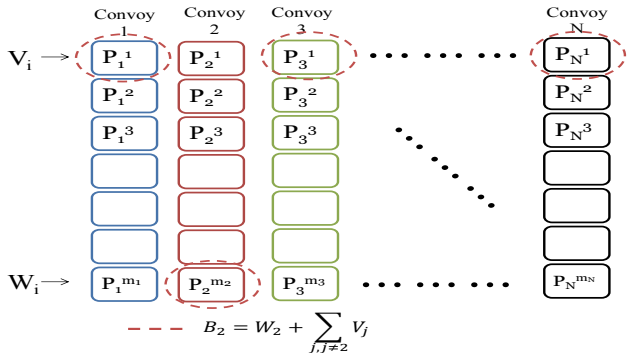

Figure 2: Best Case Scenario for Convoy 2.

Algorithm 3 Best Choice Algorithm

Input: Set of Convoys
Output: Set of pairwise compatible path

for convoy $h = 1$ to N **do** Implement 1
  if more path can be generated for h **then**
    $h_feasibility = true$
  end if
  PriorityQueue $pq \leftarrow h$ (pq holds convoys compared by feasibility and then B_i value)
end for
Stop=false;
convoy $h = pq.poll$
while Stop == False or pq is not empty **do**
  Implement Algorithm 2
  Update upper-bound
  if lower-bound¿upper-bound **then**
    Stop=true;
  end if
  if $h_feasibility = true$ **then**
    $pq.add(h)$
  end if
  $h = pq.poll()$
  if $h_feasibility = false$ **then**
    Stop=true;
  else
    Implement Algorithm 1 for h.
  end if
  if There is no more path for convoy h **then**
    $h_feasibility = false$
  end if
  update B_i value.
  update lower-bound
end while

We formalize this statement in Lemma 1 and Theorem 1. The pseudo-code of the *BestChoice* algorithm is given in Algorithm 3.

Lemma 1. The lower-bound defined in *BestChoice* algorithm is a valid lower bound for CMP.

Proof. By definition a lower-bound is a value which must be less than or equal to the objective function value of all feasible solutions. As is mentioned in section 4.2, a feasible solution is a feasible combination of paths (pairwise compatible paths). So a valid lower bound, should be always less than or equal to the objective function of all feasible combinations of paths.

The objective function is written as equation (25), for each feasible combination of paths(call it CP), a feasible path is assigned to each convoy (p_k), the objective function value will be:

$$\bar{F}^{CP}_{total} = \sum_{k=1}^{N} \sum_{(i,j)\in p^{CP}_k} \frac{l_{i,j} + l_k}{S^k_{i,j}} F_{i,j}$$

where p^{CP}_k is the feasible path assigned to convoy k in set CP of pairwise compatible paths. For simplicity, we define $cost_{p_k}$ as:

$$cost_{p_k} = \sum_{(i,j)\in p_k} \frac{l_{i,j} + l_k}{S^k_{i,j}} F_{i,j}$$

So $LB \leq \bar{F}^{CP}_{total} \quad \forall CP$, which means a valid lower-bound should be always less than or equal to $\bar{F}_{total}$, $LB \leq \sum_k cost_{p_k}$ where all p_k are pairwise compatible.

At the beginning of the *BestChoice* algorithm, the lower bound (LB_1) is just the best values of all generated paths ($LB_1 = \sum_k \{V_k\}$). Since V_k has the minimum traffic disruption for convoy k (it is the shortest path in terms of traffic disruption), $cost_{v_k} \leq cost_p$ holds for all paths p of convoy i, so $LB_1 = \sum_k v_k \leq \sum_k cost_{p_k}$, so LB_1 is a valid lower-bound.

The lower-bound for the next iterations of the algorithm are updated at the end of the previous iterations. Supposed the algorithm is in the iteration j, the lower-bound for the next iterations (LB_{j+1}) is updated iff $LB_j < UB$ (stopping criteria). Because the best combination of compatible paths of currently generated paths, if exists, is stored in UB and its value is larger than the current lower-bound, the only way that this UB can be improved is by generating more paths. The value of best case scenarios (B_i) represents the best cost combination of paths after generating more paths for convoy i, so $B_i = \sum_{k,k\neq i} v_k + w_i \leq \sum_{k,k\neq i} cost_{p_k} + cost_{p_i^{j+1}}$, where p_i^{j+1} is from the set of newly generated paths for convoy i in the iteration $j+1$. We know $LB_{j+1} = \min_i\{B_i\} \leq B_i \leq \sum_{k,k\neq i} cost_{p_k} + cost_{p_i^{j+1}}$ and it holds for each i, the cost value of a compatible combinations of paths which are generated in iteration $j+1$, if exists, is always larger or at the best situation the same as the LB_{j+1}. So in each iteration, it is not possible to have a $\bar{F}_{total}$ which its value is better than the corresponding LB in that iteration. □

Theorem 1. *BestChoice* algorithm converges to the optimal solution.

Proof. The convergence of the algorithm is shown by minimizing the distance of lower-bound and the upper-bound. By Lemma 1 it is shown that proposed LB is a valid lower-bound. In addition, an upper-bound(UB) is defined as the best found feasible solution, which is also an obvious upper-bound. We want to show that after each iteration, this gap is reduced. At the beginning, the UB is ∞ and the LB is the summation of all best generated paths values($\sum_i \{B_i\}$). Optimality gap is defined as $gap \leftarrow UB - LB$.

By the end of each iteration, two situations may happen. In case one, no feasible solution exists so far, so there is no change in UB. The updated LB (LB_{new}) is greater or equal to the previous LB (LB_{old}), because the objective value of the newly generated paths are greater or equal to the objective value of the last generated path in the previous iteration(w_i). So $UB - LB_{new} \leq UB - LB_{old}$. In case 2 a feasible solution is found, so the updated $UB(UB_{new})$ is better than previous UB (UB_{old}). We do not let the B&B algorithm branch when LB is greater than or equal

Algorithm 4 Two Convoy Algorithm

Input: lower-bound $\leftarrow 0$, upper-bound $\leftarrow +\infty$, $k1, k\_new1, k2, k\_new2 \leftarrow \{\}$
Output: updated lower-bound, possible optimal solution

```
{k1, k2} ← Generating k-best shortest paths for convoy 1 and 2 respectively(1);
generate compatibility list of all generated paths;
while stopping criteria is not met do
    if there is at least one pair of compatible paths then
        for all compatible pair(i, j)[1] in kPath do
            if value(pair(i, j)) < upper-bound then
                upper-bound ← Value(pair(i, j));
                BestSolution ← pair(i, j);
            end if
        end for
    end if
    add k_new1 to k1 and k_new2 to k2;
    lower-bound← min{value(k1_first) + value(k2_last), value(k1_last) + value(k2_first)};
    if lower-bound≥upper-bound then
        stop with BestSolution as optimal;
    end if
    generate k more path for convoy 1 and 2 and put in k_new1 and k_new2 respectively;
    if no more path can be generated then
        if upper-bound< +∞ then
            stop current BestSolution as optimal;
        else
            stop due to infeasible problem;
        end if
    else
         Generate Compatibility Matrix
        if BestSolution is not empty:(ibest, jbest) then
            compare all paths in k_new1 with paths {1 to jbest} from k2;
            compare all paths in k_new2 with paths {1 to ibest} from k1;
        else
            compare all paths in k_new1 with paths in k2 and k_new2;
            compare all paths in k_new2 with paths in k2 and k_new2;
        end if
    end if
end while
```

to UB. In addition, as mentioned earlier, the LB at each iteration has a larger value or remains unchanged. Thus $UB_{new} - LB_{new} \leq UB_{old} - LB_{old}$. In both cases the optimality gap is decreased or in the worst case, it remains the same.

We have established that the optimality gap is potentially reduced at each iteration. Due to the finite number of path choices for each convoy, the proposed method will converge to optimal solution (when gap=0) in finite number of steps. □

4.4 Special Case of Two Convoys

The special case of two convoys are studied here. When there are two convoys, we do not need to use the k-clique problem. All compatible pair of paths (one from each convoy) should be considered to see if the upper-bound can be improved. In order to do this, a compatibility matrix of all valid paths should be created.

The compatibility matrix is a boolean matrix, and its dimension is determined by the total number of feasible generated paths for each convoy. Supposed that there are 20 paths generated

[1] Pair(i, j) consists of path i of convoy1 and path j of convoy2.

for the first convoy and 24 paths generated for the second convoy, the dimension of the matrix will be 20×24. Its component in row i and column j is *true* if the path in the i^{th} row is compatible with the path in the j^{th} column, and *false* otherwise. The algorithm for solving the two-convoy case problem is presented in Algorithm 4.

Corollary 1. *TwoConvoyAlgorithm* converges to an optimal solution.

Proof. This algorithm is a special case of *BestChoice* algorithm, UB is defined as the best feasible solution found in the process and LB is the minimum possible largest value for the best combination of compatible paths, which is a valid LB based on Theorem 1. During the algorithm, LB is increased and UB is decreased. Eventually the gap $(UB - LB)$ will tend to zero (if the problem is feasible) or it will declare the problem to be infeasible. □

Although the two-convoy movement algorithm presented in Gopalan (2015) were proposed for particle convoys, zero-length convoys with no overtaking constraint and on undirected graph, so the problem basically is different than ours. The blocking of two particle convoys on directed graph may happen more than once on a path, so this algorithm cannot be applied directly here.

5 Computational Results

Since the problem solved in this paper has never been fully studied before, a direct comparison to a previously offered algorithm for CMP is not pursued. In order to study the performance and the nature of the problem, several tests have been performed on the problem. All computational studies presented in this section are implemented with Java (Jre8) and Cplex1263 (IBM, ILOG) and run on an Intel Xeon, dual cores (2.40 GHz) processor with 32 GIG of RAM with Windows7 Enterprise as operating system.

We first present a pseudo transportation network generator in subsection 5.1. We tested our proposed *BestChoice* algorithm in two ways. First by using the proposed Branch and Bound algorithm and then by manipulated Cplex for solving minimum k-clique in k-partite graph. The *BestChoice* algorithm performance measures are presented in subsection 5.2. Then in subsection 5.3 impacts of different parameters of the problem on the final solution are studied to develop policy implications.

5.1 Generating Pseudo Transportation Network Test Problems

To ensure the broad testing of our procedure, a total of six different sized transportation networks as shown in Table 3 are generated. Transportation networks are labeled as $N1$ through $N6$. Transportation networks $N1$ to $N5$ are generated by modification on networks represented in Li and Kollios (2007), one additional network with higher density is also provided as $N6$ from modification on Chicago Sketch Network (Bar-Gera 2013). In all of these cases, the connectivity of the graph was guaranteed and tested by the Kosaraju algorithm (Cormen et al. 2001). Details for each networks are available at the following URL: http://www.acsu.buffalo.edu/~batta/Convoy_Test_Instances.zip. The average degree of these test problems are presented in the Network Density column, while the min and max out degree node of these networks are from 1 to 6.

Table 3: Test Problem Networks.

Network ID	Node No.	Arc No.	Network Density	Length				Traffic Flow			
				Min	Max	Mean	STD	Min	Max	Mean	STD
N1	80	162	2.03	0.16	218.39	12.45	45.73	50.63	248.62	151.57	58.13
N2	191	399	2.09	0.15	211.58	3.04	14.97	50.20	249.91	150.03	56.12
N3	800	1957	2.45	1.50	262.52	94.46	48.33	50.06	249.68	151.64	58.13
N4	1222	2632	2.15	0.13	278.45	5.78	21.68	50.14	250.00	152.32	57.50
N5	4296	10625	2.47	0.03	362.44	8.92	32.09	50.02	249.98	149.51	57.83
N6	933	2950	3.16	0.25	158.62	11.49	10.02	50.12	249.83	151.09	57.5

The number of convoys for each test problem depends on the size of the network and varies between two, three and half or five percent of the size of the network (number of nodes). The length of a convoy varies from 0.3 to 50 miles, consistent with Marine Corps (2005). The speed of convoys for each road of the network varies from 30 mph to 50 mph and headway is defined as 30 minutes. All convoys are available at time zero, the waiting interval at the origin varies for each test problem between three, six and 12 hours. Three different values are also considered for due date, and it is shown as a coefficient of the time of the shortest path of each OD pair (each convoy). The value of this coefficient is considered as two (very tight), three and half and five (very loose). For two percent of arcs in our test problems, a tabu list of all of its neighbor arcs (where the distance to the mid point of the road is within 1 mile) is defined, so that when one of these arcs is occupied by one convoy, no other convoy can traverse its tabu arcs in any direction at that time. At each iteration of the algorithm, 40 paths are generated by using k-shortest path algorithm, considering the civilian traffic disruption. We have tested different numbers as k here and saw that it will give better results in more cases. Graph 3 shows one example of comparison of different numbers on one test problem.

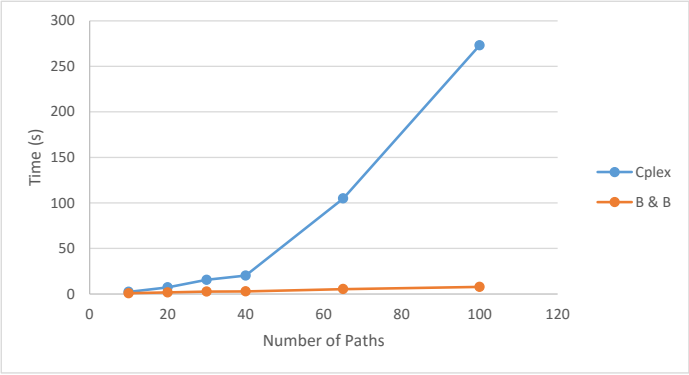

Figure 3: Tests on different path numbers

5.2 Run Time and Optimality Gap

To demonstrate the performance of the proposed algorithm, 135 test problems are generated by using five networks ($N1$ to $N5$) with different sizes and similar structure (network density) from Table 3 and by considering different values for number of convoys, the length of the waiting intervals and due dates. For each test problem, 20 different runs were performed by varying the O-D pair,

length and speeds for each convoy. We compare the results of our proposed method, with the alternative approach that uses Cplex for solving model $\mathcal{Q}$ presented in section 4.2. In the Cplex version, we added a branch call back to compare lower bound and upper bound of the current branch to see if it can be pruned. In the rest of the paper, we just list Cplex and $B\&B$ as two different methods of solving the minimum weighted k-clique in k-partite graph.

Tables 4, 5 and 6 provide detailed computational results for small, medium and large sized test problems. For each of these tables the following statistic are reported: average, min and max of 20 runs for run time and relative optimality gap[2] , the average time till finding the first feasible solution and its optimality gap and feasibility ratio[3]. A maximum of 900 second CPU time is used for each run.

Table 4 shows the performance of the proposed algorithm using either $B\&B$ or Cplex on small size problems. Either method is able to identify infeasibility or solve to optimality within 130 seconds. We further note that using $B\&B$ takes less time, on average.

Table 5 presents the performance of both methods, $B\&B$ and Cplex, for medium size transportation networks. For $N3$, both methods were able to find feasible solutions for all test problems and $B\&B$ in 17 out of 27 test problems finds the optimal solutions earlier than Cplex. For $N4$, Cplex finds more feasible solutions in test problems with longer waiting intervals and smaller due date coefficients. For problem instances with smaller due date coefficient and shorter waiting intervals, $B\&B$ finds solutions with better optimality gap as compared to Cplex.

The results of our large size transportation network, $N5$, is presented in Table 6. In this set of test problems Cplex finds more feasible solution on tighter test problems, while $B\&B$ finds more feasible and high quality solutions in test problems with loose due dates and shorter waiting intervals.

A summary table of the result for all sizes and different characteristics of the problem is represented in Table 7. Based on this table, on average, the $B\&B$ finds solutions with better quality within less amount of time, provided it can find a feasible solution. Using Cplex has a slightly higher feasibility ratio (81% versus 76%) than the $B\&B$ algorithm, on average.

For networks $N1$, $N2$ and $N3$, a feasible solution or declaration of problem infeasibility is achieved by both methods. For $N4$ and $N5$, Cplex finds feasible solutions at a better rate (67% versus 56%) more often than $B\&B$. For $N4$ test problems, using $B\&B$ finds better solutions compared to Cplex, on average. But for $N5$ test problems, Cplex results in better solutions at the end of the planning time horizon, on average.

When there are more options for each convoy (loose due date and smaller waiting time interval, more number of convoys on the same graph) using $B\&B$ results in better solutions in shorter time. The reverse is true for problems with few options for each convoy (tight due dates, larger waiting time interval, and fewer number of convoys on the same graph) with Cplex dominating.

[2]Relative optimality gap: $\frac{UB-LB}{LB} \times 100$.

[3]Feasibility ratio on each test problem is the percentage of time that a method can find a feasible solution for that problem.

Table 4: Run time and Optimality Gap for small size problems

				B&B			Cplex			B&B			Cplex			B&B		Cplex		B&B	Cplex
Network	Waiting	Number of	Due date	Optimality Gap			Optimality Gap			Time			Time			First Feasible		First Feasible		Feasibility	Feasibility
Number	Interval	Convoy (Ratio)	Coefficient	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Gap	Time	Gap	Time	Ratio	Ratio
N1	3	0.02	2	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.20%	0	0.20%	0.002	100%	100%
			3.5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.20%	0.001	0.20%	0.001	100%	100%
			5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.20%	0.002	0.20%	0.001	100%	100%
		0.035	2	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.17%	0.002	0.17%	0.002	100%	100%
			3.5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.18%	0.002	0.18%	0.002	100%	100%
			5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.16%	0.003	0.16%	0.002	100%	100%
		0.05	2	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.17%	0.003	5.29%	0.002	100%	100%
			3.5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.03%	0.003	0.03%	0.004	100%	100%
			5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.03%	0	0.03%	0.009	100%	100%
	6	0.02	2	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.20%	0	0.20%	0.001	100%	100%
			3.5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.20%	0	0.20%	0.004	100%	100%
			5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.20%	0	0.20%	0.002	100%	100%
		0.035	2	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.23%	0.002	0.23%	0.001	100%	100%
			3.5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.18%	0.002	0.18%	0.001	100%	100%
			5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.16%	0	0.16%	0.002	100%	100%
		0.05	2	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.16%	0.00	7.27%	0.003	90%	90%
			3.5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.03%	0	1.64%	0.002	100%	100%
			5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.03%	0.002	0.03%	0.003	100%	100%
	12	0.02	2	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.20%	0	0.20%	0	100%	100%
			3.5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.20%	0	0.20%	0.001	100%	100%
			5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.20%	0	0.20%	0	100%	100%
		0.035	2	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.33%	0	0.33%	0	30%	30%
			3.5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.14%	0.001	0.14%	0.002	100%	100%
			5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.18%	0	0.18%	0	100%	100%
		0.05	2	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.04%	0	0.04%	0	5%	5%
			3.5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	1.29%	0.001	6.38%	0.002	100%	100%
			5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0	0.35%	0.001	11.67%	0.003	100%	100%
N2	3	0.02	2	0%	0%	0%	0%	0%	0%	0.02	0	0.2	0.58	0.1	2.3	0.05%	0.06	0.1%	0.45	100%	100%
			3.5	0%	0%	0%	0%	0%	0%	0.005	0	0.1	1.68	0.5	3	0.1%	0.05	0.1%	1.00	100%	100%
			5	0%	0%	0%	0%	0%	0%	0	0	0	1.28	0.4	2.6	0.1%	0.03	0.1%	0.84	100%	100%
		0.035	2	0%	0%	0%	0%	0%	0%	0.025	0	0.2	2.54	0.9	4.5	0.8%	0.05	6.0%	1.26	100%	100%
			3.5	0%	0%	0%	0%	0%	0%	0	0	0	5.95	1.8	10	0.0%	0.05	0.0%	2.31	100%	100%
			5	0%	0%	0%	0%	0%	0%	0	0	0	9.81	4.2	13.9	0.0%	0.04	0.0%	4.30	100%	100%
		0.05	2	0%	0%	0%	0%	0%	0%	2.91	0	16.4	14.59	2.9	30.3	6.2%	0.19	2.8%	5.54	75%	75%
			3.5	0%	0%	0%	0%	0%	0%	0	0	0	19.00	6.4	24.6	0.2%	0.06	2.7%	7.63	100%	100%
			5	0%	0%	0%	0%	0%	0%	0.005	0	0.1	30.22	17.6	42.1	0.0%	0.07	0.0%	10.79	100%	100%
	6	0.02	2	0%	0%	0%	0%	0%	0%	0	0	0	0.02	0	0.1	5.0%	0.02	5.7%	0.07	100%	100%
			3.5	0%	0%	0%	0%	0%	0%	0	0	0	0.12	0.1	0.2	0.1%	0.02	0.1%	0.14	100%	100%
			5	0%	0%	0%	0%	0%	0%	0	0	0	0.22	0.1	0.3	0.1%	0.02	0.1%	0.23	100%	100%
		0.035	2	0%	0%	0%	0%	0%	0%	0.29	0	0.6	2.35	0.1	4.2	15.4%	0.04	21.2%	0.45	55%	55%
			3.5	0%	0%	0%	0%	0%	0%	0.025	0	0.5	0.80	0.5	1.1	2.0%	0.04	5.3%	0.70	100%	100%
			5	0%	0%	0%	0%	0%	0%	0	0	0	1.39	0.7	1.9	0.0%	0.04	0.0%	1.19	100%	100%
		0.05	2	0%	0%	0%	0%	0%	0%	8.10	0	0.6	92.10	0.3	14.1	1.0%	0.04	39.0%	0.32	5%	5%
			3.5	0%	0%	0%	0%	0%	0%	11.88	0	127.9	2.97	0.9	11.3	10.9%	6.04	22.4%	1.92	95%	95%
			5	0%	0%	0%	0%	0%	0%	4.19	0	62.5	3.68	2	5.4	1.7%	0.06	10.1%	3.17	100%	100%
	12	0.02	2	0%	0%	0%	0%	0%	0%	0	0	0.1	0.27	0	0.5	4.7%	0.02	5.1%	0.09	60%	60%
			3.5	0%	0%	0%	0%	0%	0%	0	0	0	0.10	0	0.2	6.6%	0.03	7.3%	0.11	100%	100%
			5	0%	0%	0%	0%	0%	0%	0	0	0	0	0	0.2	0.1%	0.02	0.1%	0.11	100%	100%
		0.035	2	-	-	-	-	-	-	2.40	0	0.2	21.80	0.5	2.1	-	-	-	-	0%	0%
			3.5	0%	0%	0%	0%	0%	0%	0.80	0	1	3.22	0.2	3.1	10.5%	0.07	15.4%	0.45	45%	45%
			5	0%	0%	0%	0%	0%	0%	0.11	0	2.1	0.53	0.2	1.6	2.8%	0.04	6.5%	0.50	100%	100%
		0.05	2	-	-	-	-	-	-	3.40	0	0.3	32.90	0.7	3.7	-	-	-	-	0%	0%
			3.5	0%	0%	0%	0%	0%	0%	14.00	0.3	1.8	75.30	1.1	7	33.4%	0.06	36.4%	0.58	5%	5%
			5	0%	0%	0%	0%	0%	0%	18.69	0	25.9	8.00	0.8	8.5	19.2%	0.28	24.3%	1.26	50%	50%

Table 5: Run time and Optimality Gap for medium size problem

				B&B			Cplex			B&B			Cplex			B&B		Cplex		B&B	Cplex
Network	Waiting	Number of	Due date	Optimality Gap			Optimality Gap			Time			Time			First Feasible		First Feasible		Feasibility	Feasibility
Number	Interval	Convoy (Ratio)	Coefficient	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Gap	Time	Gap	Time	Ratio	Ratio
N3	3	0.02	2	0.16%	0%	1.72%	0%	0%	0%	190.4	0.8	900	16.2	9.6	24.9	0.21%	2.6	1.24%	9.3	100%	100%
			3.5	0%	0%	0%	0%	0%	0%	1.2	0.8	2.6	330.6	163	449.2	0%	2.7	0%	148.2	100%	100%
			5	0%	0%	0%	0.0001%	0%	0.002%	1.4	0.8	2.9	689.4	517.6	1204.4	0%	2.7	0%	399.7	100%	100%
		0.035	2	0.57%	0%	2.89%	0%	0%	0%	632.2	1.7	900	55.7	23.7	119.4	1.01%	6.3	2.10%	80.2	100%	100%
			3.5	0%	0%	0%	0%	0%	0%	2.3	1.5	3.9	915.7	683	1115.7	0%	5.8	25%	589.7	100%	100%
			5	0%	0%	0%	0.001%	0.001%	0.001%	2.6	1.6	7.6	1613.6	1285.6	1989.2	0%	23.3	0%	1408.6	100%	100%
		0.05	2	2.05%	0%	6.20%	0.22%	0%	4.30%	863.7	173.2	900	259.3	72.1	946.3	4.17%	6.9	5.46%	261.2	100%	100%
			3.5	0.01%	0%	0.19%	0.48%	0%	3.52%	141.6	2.3	900	1454.1	1289.1	1602.4	0.00%	3.1	0.48%	1454.1	100%	100%
			5	0%	0%	0%	0.09%	0%	1.77%	3.7	2.5	8.1	2370.2	2142.5	2656	0%	3.3	0%	2370.2	100%	100%
	6	0.02	2	0%	0%	0%	0%	0%	0%	6.3	0.8	85.8	6.3	2.3	11.5	0%	1.6	18%	8.1	100%	100%
			3.5	0%	0%	0%	0%	0%	0%	1.2	0.8	2.4	75.0	41.2	108.4	0%	2.2	0%	37.1	100%	100%
			5	0%	0%	0%	0%	0%	0%	1.3	0.8	2.6	163.8	122.9	239.9	0%	2.5	0%	104.6	100%	100%
		0.035	2	0.66%	0%	3.22%	0%	0%	0%	800.2	1.7	900	18.0	8.2	34.6	2.17%	3.6	11.24%	17.4	100%	100%
			3.5	0%	0%	0%	0%	0%	0%	2.8	1.5	14.5	217.1	133.3	273.3	0%	3.8	0%	151.9	100%	100%
			5	0%	0%	0%	0%	0%	0%	2.5	1.5	3.8	504.7	445.8	601.4	0%	4.9	0%	302.7	100%	100%
		0.05	2	3.20%	0.78%	7.50%	0.012%	0%	0.234%	900	900	900	108.4	17.7	912.1	4.27%	5.4	8.30%	71.1	100%	100%
			3.5	0.0355%	0%	0.57%	0%	0%	0%	92.6	2.1	900	486.5	339.7	762.7	0.03%	12.9	50.19%	260.5	100%	100%
			5	0%	0%	0%	0.04%	0%	0.7%	3.3	2.2	7	1210.8	920.4	1394.5	0%	6.3	0%	709.3	100%	100%
	12	0.02	2	0%	0%	0%	0%	0%	0%	3.7	0.7	14.5	3.1	1.3	11.4	0%	2.0	9%	4.3	100%	100%
			3.5	0%	0%	0%	0%	0%	0%	1.2	0.8	3	20.4	14.1	33.1	0%	1.9	0%	16.6	100%	100%
			5	0.314%	0%	2.081%	0%	0%	0%	1.2	0.8	2.4	49.0	37.3	62.5	0.002%	2.5	0.002%	31.3	100%	100%
		0.035	2	0.31%	0%	2.08%	0%	0%	0%	328.8	1.4	900	8.6	4.9	34.9	1.46%	3.0	18.57%	13.4	100%	100%
			3.5	0%	0%	0%	0%	0%	0%	2.3	1.4	4.2	59.4	44.9	76.6	0%	6.4	0%	32.7	100%	100%
			5	0%	0%	0%	0%	0%	0%	1.9	1.4	4.7	134.3	104.2	169.4	0%	4.5	0%	89.7	100%	100%
		0.05	2	1.92%	0%	4.25%	0%	0%	0%	823.5	187.7	900	79.2	12	469.6	5.57%	5.2	16.88%	38.1	100%	100%
			3.5	0.011%	0%	0.123%	0%	0%	0%	183.3	2.2	900	135.9	118.4	165.4	0.233%	6.8	0.207%	193.9	100%	100%
			5	0%	0%	0%	0%	0%	0%	4.8	2.1	20.3	304.6	234.1	370.1	0%	4.8	0%	139.5	100%	100%
N4	3	0.02	2	1.00%	0%	2.90%	0%	0%	2.38%	835.07	12.4	900	225.2	12.8	906.3	4.28%	5.3	0.71%	43.4	60%	100%
			3.5	0%	0%	0%	0%	0%	0%	2.045	1.6	2.5	75.7	43.2	94.3	0%	4.6	12%	70.9	100%	100%
			5	0%	0%	0%	0%	0%	0%	1.865	1.6	2.6	143.6	105.5	197.3	0%	6.5	0%	132.0	100%	100%
		0.035	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%
			3.5	0.90%	0%	3.06%	0%	0%	3.01%	640.08	2.8	900	404.5	252.6	998.2	1%	3.7	75.45%	512.8	100%	100%
			5	0%	0%	0%	0%	0%	0%	7.135	3	54	561.7	413.2	707.6	0%	7.5	51%	340.6	100%	100%
		0.05	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%
			3.5	5.15%	3.17%	6.86%	7.58%	0%	15.16%	900.005	900	900.1	1074.1	958.6	1145.4	36%	109.9	100%	708.8	25%	90%
			5	0%	0%	1.22%	5.47%	0%	12.27%	500.69	4.6	900	1296.0	1095.3	1373.4	0.32%	10.7	1.8%	857.2	100%	80%
	6	0.02	2	1.54%	0%	3.38%	3.26%	0%	9.80%	812.52	3.7	900	646.4	5.5	911.3	1.45%	3.97	1.85%	19.5	25%	90%
			3.5	0%	0%	0%	0%	0%	0%	100.855	1.6	900	14.8	9.6	20.2	0.57%	6.0	14%	25.8	100%	100%
			5	0%	0%	0%	0%	0%	0%	1.795	1.6	2.4	26.5	20.4	33.1	0%	5.4	0%	41.8	100%	100%
		0.035	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%
			3.5	2.47%	0%	6.99%	0%	0%	0%	900	900	900	186.1	55.7	923.8	5%	11.9	2.12%	137.2	70%	100%
			5	0%	0%	0.55%	0%	0%	0%	274.45	3	900	122.1	86.4	169.1	0%	9.4	15.33%	226.5	95%	100%
		0.05	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%
			3.5	-	-	-	4.30%	0%	10.15%	900	900	900	944.6	922.1	964.8	-	-	53%	273.4	0%	40%
			5	1.64%	0%	4.97%	0%	0%	2.71%	855.23	4.4	900.2	427.6	182.7	1024.6	1%	8.2	0.54%	518.1	90%	100%
	12	0.02	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%
			3.5	1.85%	0%	8.77%	0%	0%	1.16%	573.5	15.1	900	61.3	4.5	901.4	7%	5.0	16.94%	15.3	80%	100%
			5	0%	0%	1.42%	0%	0%	0%	121.965	1.6	900	8.9	7.1	18.1	0.931%	5.7	29.35%	23.3	100%	100%
		0.035	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%
			3.5	-	-	-	9.23%	5.00%	13.17%	-	-	-	904.6	895.2	909.9	-	-	55%	171.3	0%	25%
			5	3.35%	1.17%	5.56%	0%	0%	6.43%	900	900	900	219.6	26.3	925.9	6%	8.2	3.27%	67.4	50%	95%
		0.05	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%
			3.5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%
			5	-	-	-	6.06%	1.81%	10.28%	900	900	900	926.8	898.9	944.6	-	-	51%	330.4	0%	70%

Table 6: Run time and Optimality Gap for large size problems

				B&B			Cplex			B&B			Cplex			B&B		Cplex		B&B	Cplex
Network Number	Waiting Interval	Number of Convoy (Ratio)	Due date Coefficient	Optimality Gap			Optimality Gap			Time			Time			First Feasible		First Feasible		Feasibility Ratio	Feasibility Ratio
				Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Gap	Time	Gap	Time		
N5	3	0.02	2	2.20%	1%	3.90%	0%	0%	0.51%	900	900	900	556.3	87.6	929.8	7.10%	91.88	7.42%	166.14	95%	100%
			3.5	0%	0%	0%	0%	0%	0%	47.3	38.5	54.6	260.6	180.5	285	0%	46.923	5%	204.2	100%	100%
			5	0%	0%	0%	0%	0%	0%	37.46	35.1	39.2	449.6	405.3	513.8	0%	44.7072	0%	401	100%	100%
		0.035	2	4%	4%	4%	3.65%	1.24%	5.93%	902	900	908	1383.7	961.2	1487.6	7%	167.595	4%	637.76	5%	80%
			3.5	0.24%	0%	0.51%	0%	0%	0.00%	900	900	900	978.9	852.4	1037.4	0.42%	79.82	4.66%	967.90	100%	100%
			5	0%	0%	0%	0%	0%	0%	120.215	68.9	900	1765.5	1611.6	1960.6	0%	81.51	0.13%	1165.55	100%	90%
		0.05	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%
			3.5	1.04%	0.49%	1.73%	-	-	-	900	900	900	1851.3	1801.2	1920.2	1.25%	128.183	-	-	100%	0%
			5	0%	0%	0.30%	0.54%	0%	0.90%	705.935	106.8	900	3347.4	3136.2	3688.4	0.09%	171.57	0.09%	2473.35	100%	25%
	6	0.02	2	3.96%	2%	5.77%	0.96%	0%	2.11%	900	900	900	904.9	896	916.3	4.99%	41.06	12.76%	74	55%	95%
			3.5	0%	0%	0%	0%	0%	0%	523.905	34.9	900	83.4	76.6	92.4	0%	52.19	13%	108.4	100%	100%
			5	0%	0%	0%	0%	0%	0%	40.875	35.8	54.2	135.9	125	159.6	0%	45.46	0%	139.66	100%	100%
		0.035	2	-	-	-	6%	4%	9%	901	900	906	1006.1	904.7	1048.9	-	-	81%	903.84	0%	35%
			3.5	0.89%	0%	1.94%	0%	0%	0%	903	900	907	950.3	425.4	1121	1.02%	78.17	3.87%	284.8	100%	100%
			5	0%	0%	0.26%	0%	0%	0%	572.465	159.5	903	1997.2	715.6	2662.6	0.02%	79.30	17.96%	415.90	100%	100%
		0.05	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%
			3.5	3%	2%	4%	0.48%	0%	0.86%	900	900	900	1152.1	967.7	1185.5	2.58%	115.816	0%	598	65%	100%
			5	0.25%	0%	0.67%	0%	0%	0.00%	900	900	900	985.6	899.6	1087.5	0.26%	111.39	0.34%	965.4	100%	100%
	12	0.02	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%
			3.5	1.12%	0%	2.22%	0%	0%	0.54%	900	900	900	339.2	47.8	913.4	1.12%	42.86	15.21%	71.85	90%	100%
			5	0%	0%	0.55%	0%	0%	0%	700.555	36.1	900	76.8	64.5	216.3	0.17%	45.82	13.11%	80.09	100%	100%
		0.035	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%
			3.5	5%	3%	6%	1.47%	0.41%	2.55%	901	900	904	969	902.9	1033.9	5%	74.91	6%	485.89	35%	85%
			5	0.94%	0.35%	1.64%	0%	0%	0.33%	901	900	905	885	663.3	1152.8	0.95%	76.00	0.30%	190.8	100%	100%
		0.05	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0%	0%
			3.5	-	-	-	5%	4%	7%	-	-	-	971.04	911.6	993.6	-	-	10%	97.53	0%	60%
			5	2%	1%	3%	0.34%	0.03%	1.19%	900	900	900	1014.07	899.3	1096.9	2.02%	120.18	1.1%	329.2	60%	100%

Another interesting conclusion from our summary table is that, using $B\&B$ finds a feasible solution (with acceptable optimality gap) earlier than Cplex in all the cases where both algorithms find a feasible solution. An example of the comparison of this property is shown in Figure 4. This graph is a result of one of the runs of a specific problem and shows the time and quality of the feasible solutions found with both methods during their run times. The capability of $B\&B$ for finding feasible solutions faster than Cplex makes it an appealing function to be used for creating MIP start for Cplex. In most of the medium and large test problems, the pre-processing step of Cplex was highly time consuming. In some cases it was more than our time limit, and that is the reason the run time of Cplex in those cases is higher than 900 seconds.

Table 7: Summary Table of Results

		B&B					Cplex				
		Gap (%) Avg	Time(S) Avg	First Feasible Gap	First Feasible Time	Feasibility Ratio	Gap (%) Avg	Time(S) Avg	First Feasible Gap	First Feasible Time	Feasibility Ratio
Total	Avg	0.4%	239.6	2.1%	17.84	76%	0.5%	343.9	9.2%	203.91	81%
Network Number	N1	0.0%	0.00	0.2%	0.00	93.5%	0.0%	0.00	1.3%	0.00	93.5%
	N2	0.0%	2.48	4.8%	0.30	77.4%	0.0%	12.28	8.4%	1.82	77.4%
	N3	0.3%	185.18	0.7%	5.07	100%	0.0%	418.15	6.2%	331.23	100%
	N4	1.2%	512.62	4.0%	13.25	48%	1.9%	435.26	25.4%	237.66	65%
	N5	1.2%	688.46	1.7%	84.77	63%	0.9%	1002.90	9.4%	512.44	69%
Convoy No	0.02	0.29%	155.99	1%	11.03	90%	0.11%	124.82	4.40%	55.34	94%
	0.035	0.49%	265.06	2%	19.47	75%	0.53%	388.04	10.83%	230.14	83%
	0.05	0.60%	309.23	4%	24.63	64%	0.86%	544.17	13.07%	352.23	66%
Duedate Coefficient	2	0.70%	297.49	3%	11.56	52%	0.48%	164.98	9.41%	76.00	60%
	3.5	0.51%	248.73	3%	19.81	82%	0.67%	341.49	13.06%	177.53	88%
	5	0.21%	188.61	1%	20.29	94%	0.29%	477.45	5.33%	317.25	96%
Waiting Interval	3	0.41%	220.05	2%	24.23	88%	0.45%	527.73	7.65%	376.51	88%
	6	0.43%	269.55	2%	15.44	81%	0.37%	297.09	9.96%	152.45	87%
	12	0.51%	227.46	3%	12.62	60%	0.64%	192.44	10.02%	67.38	69%

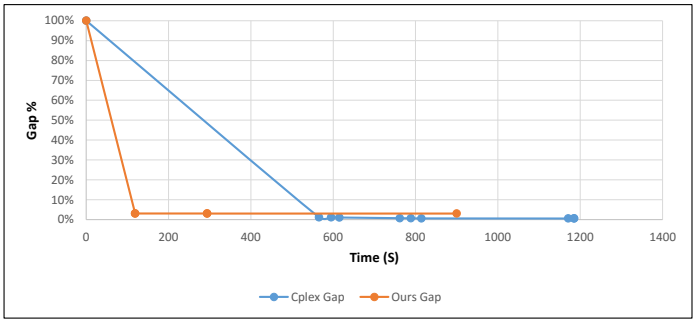

Figure 4: The comparision of finding feasible solutins for N5 with waiting interval=6, duedate co-effiecient=3.5 and convoy no=0.05

5.3 Policy Implication Study

In order to specifically study the nature of the problem, we generated test problems based on a variety of design variables such as number and size of convoys, the looseness of due dates and the length of the waiting intervals. In this section the impact of changing each of these parameters are studied.

To purely focus on the structure of the problem, the Chicago Sketch Network (N6 in our test problem) from Bar-Gera (2013) is chosen (by some modification on its arc length as the website suggested) since this has the most dense network in an attempt to increase the chance of generating feasible test problems. To support this idea in the next section 5.3.1 the effect of the network density is tested. In these set of test problems the length of each convoy is uniformly distributed between 6 to 30 miles and the runtime of the algorithm is limited to 10 minutes.

5.3.1 Effect of Graph Density

In order to see the effect of the graph density on the CMP, we test the density of a small size network of 100 nodes. Results are shown in Table 5. By having a more dense graph, the chance that a pair of convoys encounter each other on the network decreases and so the total traffic disruption is decreased.

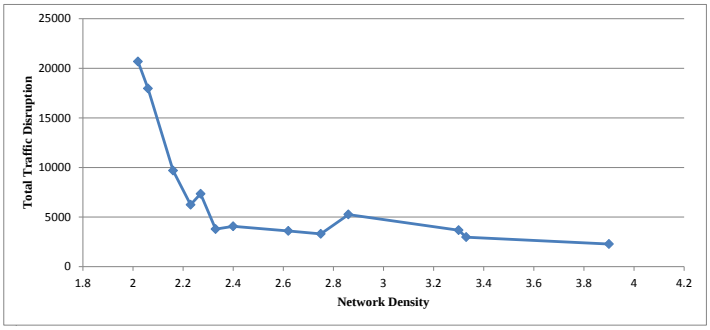

Figure 5: The Effect of Network Density.

5.3.2 Changing the Due Dates

The tightness or looseness of the due dates highly impacts the solution of the CMP. In order to study its impact, due dates are formulated by the following formula αT_{SP}, where α is a coefficient between 1 to 10 and T_{DP} is the time which a convoy need to traverse its shortest path. As we can see in Figure 6, for the case $\alpha = 1$ the problem is infeasible. In such a case, all convoys should follow their due dates, while there is no flexibility at the origin. As α gets larger, the objective value is improved, because more convoys are assigned to paths with minimum traffic disruption (which may not be the time shortest path) or by waiting at the origin. After $\alpha = 6$ the objective function will remain the same, because, all convoys can use their shortest paths with the current due date.

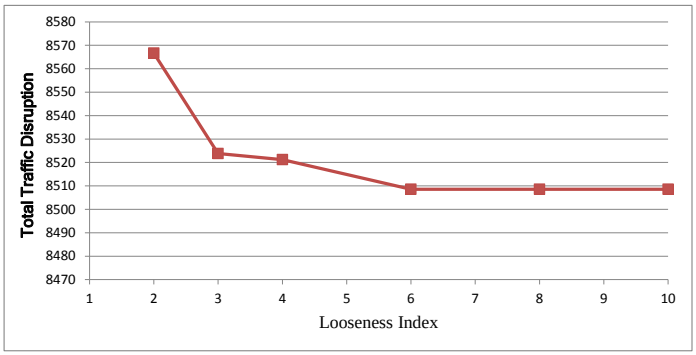

Figure 6: The Effect of Extending the Due Date.

Loose due date will improve the objective value dramatically, but at the same time it will increase the risk factor of convoys. Although we are studying CMP in peacetime, having convoys circling around on a network for a long period of time is not safe. So using a due date coefficient more than three or four is not suggested. In our problem, $\alpha = 3$ can be a good option because this provides a significant improvement to the objective value and at the same time it is not too loose. Also, after $\alpha = 3$ the change in the objective function is not so prominent.

5.3.3 Changing the Waiting Time Intervals at the Origin

Waiting time intervals specify the departure time of a convoy and can play a critical role to help convoys take their best choices while avoiding encounters with other convoys. Since waiting at the origin is not penalized, with a loose due date, convoys prefer to wait at the origin, so that they can use their best paths to arrive to their destinations. However, restricted waiting time intervals can even affect a very loose due date. The effect of extending the waiting time interval on the total traffic disruption is shown in Figure 7. The objective value of time period 1,2 and 4 are the same and have the smallest value, equal to the value of the best paths. This can be explained as follows: In all of these solutions, either convoys depart the origin at time zero or time 4 and follow their best choices. So having the smaller time window helps to catch all the optimal situations, by having a number which is a factor of most of the time intervals. In this case, intervals with length 1 are ideal. This, however increases the complexity of the problem. Choosing a suitable waiting time interval (e.g. the largest common factor) is thus very important.

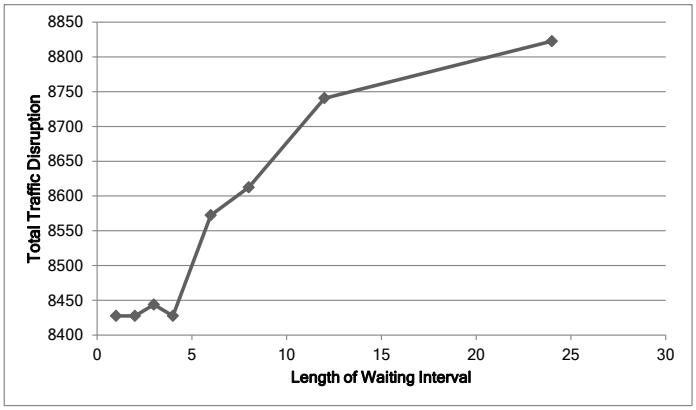

Figure 7: The Effect of Changing the Waiting Time Interval.

6 Summary, Conclusions and Future Research

We define a new objective function of a convoy movement problem, in peacetime situation, which attempts to find a routing for convoys to decrease the amount of civilian traffic disruption. Our proposed method consists of two elements, first k paths with minimum traffic disruption are generated for each convoy and then each convoy is assigned to one of these paths, while ensuring that selected paths are all pairwise compatible. The k-shortest path algorithm is used to generate paths. The minimum weighted k-clique in k-partite graph problem, solved through a modified branch and bound algorithm, is used to select pairwise compatible paths for each convoy.

The performance result of the proposed methods shows that it can solve problems up to 1000 nodes optimally in a short period of time (15 minutes) and even when the algorithm is unable to find the optimal solution in 15 minutes for large size problem, the quality of the best feasible solution is very good. The impact of changing different parameters of the problem on final results are studied. This provides a better insight to the CMP. In addition, the solution of the proposed $B\&B$ can be used as an starting point for Cplex in solving minimum k-clique in k-partite graph, due to its capability of finding feasible solutions with high quality faster than Cplex.

We assume discrete time intervals for convoys as the waiting time at the origin. Continuous waiting time intervals would be an interesting extension. Also, application of our method to other versions of the disjoint paths problem can be considered. Traffic volume can be considered as dynamic, time dependent, to improve modeling realism. Finally, the convoy formation problem (which decides the size and frequency of convoy movement) is of much relevance and is suggested as future research.

References

Bar-Gera, H. (2001-2013). Transportation network test problems.

Bellman, R. (1958). On a routing problem. *Quart. of Applied Math. 16*, 87–90.

Bovet, J., C. Constantin, and D. de Werra (1991). A convoy scheduling problem. *Discrete Applied Math. 30*(1), 1–14.

Chardaire, P., G. P. McKeown, S. A. Verity-Harrison, and S. B. Richardson (2005). Solving a time-space network formulation for the convoy movement problem. *Oper. Res. 53*(2), 219–230.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, C. Stein, et al. (2001). *Introduction to algorithms*, Volume 2. MIT Press Cambridge, MA.

Dreyfus, S. E. (1969). An appraisal of some shortest-path algorithms. *Oper. Res. 17*(3), 395–412.

Eppstein, D. (1998). Finding the k shortest paths. *SIAM J. Comput. 28*(2), 652–673.

Ford Jr, L. R. (1956). Network flow theory. Technical report, The Rand Corporation, Santa Monica, CA.

Goldstein, D., T. Shehab, J. Casse, and H. Lin (2010). On the formulation and solution of the convoy routing problem. *Transportation Res. E: Logistics and Transportation Review 46*(4), 520–533.

Gopalan, R. (2015). Computational complexity of convoy movement planning problems. *Mathematical Methods of Operations Research*, 1–30.

Gopalan, R. and N. S. Narayanaswamy (2009). Analysis of algorithms for an online version of the convoy movement problem. *J. of the Operational Res. Society 60*(9), 1230–1236.

Grünert, T., S. Irnich, H. Zimmermann, M. Schneider, and B. Wulfhorst (2002). Finding all k-cliques in k-partite graphs, an application in textile engineering. *Computers & Oper. Res. 29*(1), 13–31.

Harrison, S. A. (2001). Convoy planning in a digitized battlespace. Technical report, Defence Evaluation and Research Agency Malvern (United Kingdom).

Kumar, P. N. R. and T. T. Narendran (2010). Convoy movement problem–an optimization perspective. In *Innovations in Defence Support Systems–1*, pp. 79–93. Springer.

Kumar, P. N. R. and T. T. Narendran (2011). On the usage of lagrangean relaxation for the convoy movement problem. *J. of the Operational Res. Society 62*(4), 722–728.

Lee, Y. N., G. P. McKeown, and V. J. Rayward-Smith (1996). The convoy movement problem with initial delays. *Modern Heuristic Search Methods*, 215–236.

Li, F. and G. Kollios (2007). Real datasets for spatial databases: Road networks and points of interest.

Marine Corps, U. (2005). *Convoy Operations Handbook*. Honolulu, Hawaii : University Press of the Pacific.

Martins, E. Q. V. and J. L. E. Dos Santos (1996). A new shortest paths ranking algorithm. Technical report, Department of Mathematics, Univeristy of Coimbra.

Martins, E. Q. V. and M. M. B. Pascoal (2003). A new implementation of yens ranking loopless paths algorithm. *Quart. J. of the Belgian, French and Italian Oper. Res. Societies 1*(2), 121–133.

Mirghorbani, M. and P. Krokhmal (2013). On finding k-cliques in k-partite graphs. *Optim. Lett. 7*(6), 1155–1165.

Mohanta, K. and B. P. Poddar (2012). Comprehensive study on computational methods for k-shortest paths problem. *Internat. J. of Computer Appl. 40*(14), 22–26.

Montana, D., G. Bidwell, G. Vidaver, and J. Herrero (1999). Scheduling and route selection for military land moves using genetic algorithms. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, Volume 2, pp. 1118–1123. IEEE.

Moore, E. (1959). *The Shortest Path Through a Maze*. Bell Telephone System. Technical publications. monograph. Bell Telephone System.

Pollack, M. (1961). Letter to the editor-the k th best route through a network. *Oper. Res. 9*(4), 578–580.

Robinson, . M. and E. L. Leiss (2006). Applying genetic algorithms to convoy scheduling. In *Artificial Intelligence in Theory and Practice*, pp. 315–323. Springer.

Thangarajoo, R. and H. C. Lau (2010). Distributed route planning and scheduling via hybrid conflict resolution. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, Volume 2, pp. 374–378. IEEE.

Tuson, A. L. and S. A. Harrison (2005). Problem difficulty of real instances of convoy planning. *J. of the Operational Res. Society 56*(7), 763–775.

Vanderbeck, F. and M. W. Savelsbergh (2006). A generic view of dantzig–wolfe decomposition in mixed integer programming. *Operations Research Letters 34*(3), 296–306.

Vassilevska, V. (2009). Efficient algorithms for clique problems. *Information Processing Lett. 109*(4), 254–257.

Yen, J. Y. (1971). Finding the k shortest loopless paths in a network. *Management Sci. 17*(11), 712–716.